



This PDF is generated from authoritative online content, and is provided for convenience only. This PDF cannot be used for legal purposes. For authoritative understanding of what is and is not supported, always use the online content. To copy code samples, always use the online content.

Genesys Mobile Services Deployment Guide

Configuring and Starting a GMS Cluster

Contents

- 1 Configuring and Starting a GMS Cluster
 - 1.1 Prerequisites
 - 1.2 Introduction
 - 1.3 Initializing a Single-Node Cluster
 - 1.4 Initializing a Multi-Node or Multi-Data Center Cluster
 - 1.5 Load Balancing Between GMS Instances
 - 1.6 Limitations

Configuring and Starting a GMS Cluster

Important

Even if GMS multi-node deployment requires a Load Balancing solution, Genesys does not provide support for third-party software. The steps in this page refer to a sample deployment of HAProxy that should be used only as an example. You must deploy an adequate flavor of Load Balancing software in front of GMS.

Prerequisites

- GMS version 8.5.x
- Red Hat Linux version 5.0 (32 bit and 64 bit), 4Gb RAM or Higher (for the HA Proxy Load Balancer)
- Before 8.5.206.04: Support for JDK 8 only
- Starting in 8.5.206.04: Support for Open JDK 8
- GMS nodes and cluster are **configured**.
- External Cassandra is **configured**.

Introduction

The process for initializing a GMS cluster (whether it is a single node, multiple nodes, or multiple data center cluster) is to first correctly configure the **Node and Cluster Initialization Properties** in each node's **cassandra.yaml** configuration file, and then start each node individually, starting with the seed node(s). Configuration file **cassandra.yaml** is automatically generated by GMS Installation Package, you don't need to update the file until you need specific settings. Installation Package proposes to choose between the type of node "seed node/Not a seed node". The following section details the GMS cluster setup.

Initializing a Single-Node Cluster

GMS is intended to run on multiple nodes, however, you may want to start with a single node cluster for evaluation purposes.

To start GMS on a single node, set the following required properties in the **cassandra.yaml** file:

```
cluster_name: GMS Cluster
initial_token:
```

(Optional) The following properties are already correctly configured for a single node instance of Cassandra. However, if you plan on expanding to more nodes after your single-node evaluation, setting these correctly the first time you start the node is recommended.

```
seeds: <IP of GMS node>
listen_address: <IP of GMS node>
rpc_address: <IP of GMS node>
```

Start GMS on the node.

Initializing a Multi-Node or Multi-Data Center Cluster

To correctly configure a multi-node or multi-datacenter cluster you must determine the following information:

- A name for your cluster
- How many total nodes your cluster will have, and how many nodes per data center (or replication group)
- The IP addresses of each node
- The token for each node (see [Calculating Tokens](#)). If you are deploying a multi-datacenter cluster, make sure to assign tokens so that data is evenly distributed within each data center or replication grouping (see [Calculating Tokens for Multiple Data Centers](#)).
- Which nodes will serve as the seed nodes. If you are deploying a multi-datacenter cluster, the seed list should include a node from *each* data center or replication group.

This information will be used to configure the [Node and Cluster Initialization Properties](#) in the `cassandra.yaml` configuration file on each node in the cluster. Each node should be correctly configured before starting up the cluster, one node at a time (starting with the seed nodes). For example, suppose you are configuring a 4 nodes cluster spanning 1 rack in a single data center. The nodes have the following IPs, and one node in the rack will serve as a seed:

- GMS node 172.25.157.171 (seed)
- GMS node1 172.25.157.177
- GMS node2 172.25.157.179
- GMS node3 172.25.157.185

The `cassandra.yaml` files for each node would then have the following modified property settings.

node0

```
cluster_name: 'GMS Cluster'
initial_token:
seed_provider:
  - seeds: '172.25.157.171'
listen_address: 172.25.157.171
```

```
rpc_address: 172.25.157.171
```

node1

```
cluster_name: 'GMS Cluster'  
initial_token:  
seed_provider:  
  - seeds: '172.25.157.171'  
listen_address: 172.25.157.177  
rpc_address: 172.25.157.177
```

node2

```
cluster_name: 'GMS Cluster'  
initial_token:  
seed_provider:  
  - seeds: '172.25.157.171'  
listen_address: 172.25.157.179  
rpc_address: 172.25.157.179
```

node3

```
cluster_name: 'GMS Cluster'  
initial_token:  
seed_provider:  
  - seeds: '172.25.157.171'  
listen_address: 172.25.157.185  
rpc_address: 172.25.157.185
```

When the installation and configuration are done for all GMS's, you can start each instance.

Load Balancing Between GMS Instances

Load balancing is a computer networking methodology to distribute workload across multiple computers or a computer cluster, network links, central processing units, disk drives. In a GMS Cluster, Load Balancing is used to distribute the workload across multiple GMS instances. The software is installed on a separate Red Hat host. The installation of HAProxy is described [here](#). See also [How to setup HAProxy as Load Balancer for Nginx on CentOS 7](#). Once installed, you have to create a configuration file for HAProxy "haproxy-gms.cfg" and copy the following in the file:

```
global  
  daemon  
  maxconn 256  
defaults  
  mode http  
  timeout connect 5000ms  
  timeout client 50000ms  
  timeout server 50000ms  
frontend http-in  
  bind *:8080  
  default_backend cluster_gms  
listen admin  
  bind *:9090  
  stats enable  
backend cluster_gms  
  balance roundrobin # Load Balancing algorithm  
  #following http check, is used to know the status of a GMS (using NodeService from
```

Configuring and Starting a GMS Cluster

GMS)

```
option httpchk GET /genesys/1/node
option forwardfor # This sets X-Forwarded-For
## Define your servers to balance
server server1 172.25.157.171:8080 weight 1 maxconn 512 check
server server2 172.25.157.177:8080 weight 1 maxconn 512 check
server server3 172.25.157.179:8080 weight 1 maxconn 512 check
server server4 172.25.157.185:8080 weight 1 maxconn 512 check
```

Once done, you can start HAProxy using the following command:

```
[root@bsgenhaproxy haproxy]# ./haproxy -f haproxy-gms.cfg
```

GMS Service Management UI

Cluster view in the [GMS Service Management User Interface](#), Home page:

Last Updated: 7/31/2013 12:59:25

IP:

Token: 75046021690165490968853874128439747963
Status: Down

Load: ?
Data Center: datacenter1
Rack: rack1
Own: 14.69%
Running Since: Wed Jul 31 2013 12:59:25 GMT+0300

IP:

Token: 50057505283674829242183335979434942266
Status: Up

Load: 151.69 MB
Data Center: datacenter1
Rack: rack1
Own: 85.31%
Running Since: Tue Jul 30 2013 14:36:18 GMT+0300

HAProxy Statistics Report page

The following page is available at: http://<haproxy_host>:9090/haproxy?stats

HAProxy version 1.4.22, released 2012/08/09

Statistics Report for pid 4043

> General process information

pid = 4043 (process #1, nproc = 1)
 uptime = 0s 0h0m0s
 system limits: memmax = unlimited; ulimitn = 528
 maxsock = 528; maxconn = 250; maxpipes = 0
 current conns = 1; current pipes = 0/0
 Running tasks: 1/3

■ active UP ■ backup UP
■ active UP, going down ■ backup UP, going down
■ active DOWN, going up ■ backup DOWN, going up
■ active or backup DOWN ■ not checked
■ active or backup DOWN for maintenance (MAINT)
 Note: UP with load-balancing disabled is reported as "NO LB"

Display option: [Hide DOWN servers](#) External resources: [Primary site](#)
[Refresh now](#) [Updates \(v1.4\)](#)
[CSV export](#) [Online manual](#)

	Queue			Session rate			Sessions			Bytes		Denied		Errors		Warnings		Server													
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Downtme	Thrtle		
haproxy																															
Frontend	0	0	-	0	0	-	0	0	2 000	0	0	0	0	0	0	0	0	0	0	0	0	OPEN									
admin																															
Frontend	1	1	-	1	1	-	1	1	2 000	1	1	0	0	0	0	0	0	0	0	0	0	OPEN									
Backend	0	0	-	0	0	-	0	0	2 000	0	0	0	0	0	0	0	0	0	0	0	0	3s UP		0	0	0	0	0	0	0	
cluster_gms																															
server1	0	0	-	0	0	-	0	0	512	0	0	0	0	0	0	0	0	0	0	0	0	3s DOWN	L4CON in 0ms	1	Y	-	0	0	1	3s	-
server2	0	0	-	0	0	-	0	0	512	0	0	0	0	0	0	0	0	0	0	0	0	3s UP	L7OK:200 in 3ms	1	Y	-	0	0	0	0s	-
Backend	0	0	-	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3s UP		1	1	0	0	0	0s		

You can now use the HAProxy endpoint `http://<haproxy_host>:<haproxy_port>` as the main entry point for the Cluster.

Limitations

If you set up different clusters of GMS for different purposes (Callback, and so on), you must change the cluster name to be able to start GMS (this applies to embedded, and external Cassandra clusters if they are set up for different purposes).

Example of GMS architecture with Cassandra clusters for different purposes:

